Information Integration

Project Presentation

Giuseppe Prisco 1895709

STRUCTURE OF THE PRESENTATION

1. DATA GATHERING

- Scenario of Interest
- Collected Data
- Identified Tasks

2. MODELLING

- Source Schema
- Global Schema
- Mapping Layer
- Queries

3. IMPLEMENTATION

- Pre-processing
- Mapping Transformations
- Materialization

1

DATA GATHERING

Scenario of Interest,
Collected Data,
Identified Tasks

SCENARIO OF INTEREST

- The **domain** I chose for the development of the project is represented by **video games**
- Video games generally belong to **different platforms**, such as PC games available on Steam, Nintendo games available on the Nintendo Switch or video games available on the PlayStation 4 console
- Each year a ceremony for honoring achievements in the videogame industry is held, called "**The Game Awards**"
- In this ceremony, different nominees are chosen to compete in different categories to win an award, with the most known between them being the "**Game of the Year**" award

COLLECTED DATA

The **datasets** I chose were all taken from the **Kaggle website**. They include:

- a dataset containing **Steam games**
- a dataset containing **Nintendo games**
- a dataset containing **PlayStation 4 games**
- a dataset containing "**The Game Awards**" from 2014 to 2019

IDENTIFIED TASKS

The possible **tasks** for this kind of system can be the following:

- given a company, find all the games that have been published by that company
- find all the cross-platform games that won an award for both "Game of the Year" and "Best Action/Adventure Game"
- find all games that won at least two awards in the same year in two different categories

2

MODELLING

Source Schema,
Global Schema,
Mapping Layer, Queries

PRE-PROCESSING OF DATA

Before analyzing the source schema, the data gathered from the chosen datasets has been **pre-processed**, to discard useless information. In particular:

- Only a subset of the original attributes was considered, removing the least interesting ones
- Rows containing non UTF-8 encoded characters were identified and discarded through the use of regular expressions
- Sometimes it happened that there were double or triple instances of “ characters in a row, they have been replaced with a single “

Resident Evil 2 ““R.P.D Demo”” ⟶ Resident Evil 2 “R.P.D Demo”

Game Categories (1)

- Best VR/AR Game
- Student Game Award
- Best Music/Sound Design
- Best Strategy Game
- ESports Game of the Year
- Best Online Experience
- Best VR Game
- Best Handheld Game
- Most Anticipated Game
- Best Student Game
- Games for Change
- Player's Voice
- Best Esports Game
- Best Multiplayer
- Most Anticipated Game 2015
- Best Multiplayer Game
- Best Score/Soundtrack
- Best Action Game
- Best Fighting Game
- Best Score/Music
- Best Remaster
- Best Art Direction

Game Categories (2)

- Fresh Indie Game
- Global Gaming Citizens
- Best Audio Design
- Best eSports Game
- Best Independent Game
- Best Shooter
- Best Fan Creation
- Game of the Year
- Best Mobile/Handheld Game
- Best Sports/Racing Game
- Best Game Direction
- Best Ongoing Game
- Best Mobile Game
- Games for Impact
- Best Family Game
- Best Debut Indie Game
- Best Role Playing Game
- Best Action/Adventure Game
- Best Community Support
- Best Narrative
- Chinese Fan Game Award

People and Company Categories

People

- Developer of the Year
- Trending Gamer
- Content Creator of the Year
- Best Esports Coach
- Best Esports Host
- Best Esports Player
- Best Performance
- ESports Player of the Year
- Industry Icon Award

Company

- Best Esports Event
- Best Esports Moment
- Best eSports Team
- Best Esports Team
- ESports Team of the Year

SOURCE SCHEMA

The **source schema**, after an initial pre-processing of data, is the following:

- steamgames(id, game, release, rating, primary_genre, publisher, developer)
- switchgames(position, game, publisher, developer, total_shipped, release_date, last_update)
- ps4games(id, game, publisher, release_date, developer, genre)
- gameawards(year, category, nominee, company, winner, voted)

GLOBAL SCHEMA ALPHABET

game$_{/2}$

crossgame$_{/2}$

publisher$_{/1}$

developer$_{/1}$

hasPublished$_{/2}$

hasDeveloped$_{/2}$

award$_{/6}$

GLOBAL SCHEMA

game(name, release_date)

crossgame(name, release_date)

publisher(name)

developer(name)

hasPublished(name, game)

hasDeveloped(name, game)

award(year, category, nominee, winner, game, type)

MAPPING LAYER (1)

From **steamgames** we can take information about games, publishers and developers:

$\forall na,rel.\ \exists i,rat,ge,pub,dev.steamgames(i,na,rel,rat,ge,pub,dev) \rightarrow game(na,rel)$

$\forall pub.\ \exists i,na,rel,rat,ge,dev.steamgames(i,na,rel,rat,ge,pub,dev) \rightarrow publisher(pub)$

$\forall dev.\ \exists i,na,rel,rat,ge,pub.steamgames(i,na,rel,rat,ge,pub,dev) \rightarrow developer(dev)$

$\forall pub,na.\ \exists i,rel,rat,ge,dev.steamgames(i,na,rel,rat,ge,pub,dev) \rightarrow hasPublished(pub, na)$

$\forall dev,na.\ \exists i,rel,rat,ge,pub.steamgames(i,na,rel,rat,ge,pub,dev) \rightarrow hasDeveloped(dev, na)$

MAPPING LAYER (2)

Also from **switchgames** we can take information about games, publishers and developers:

$\forall na,rel.\ \exists i,pub,dev,tot,up.switchgames(i,na,pub,dev,tot,rel,up) \rightarrow game(na,rel)$

$\forall pub.\ \exists i,na,dev,tot,rel,up.switchgames(i,na,pub,dev,tot,rel,up) \rightarrow publisher(pub)$

$\forall dev.\ \exists i,na,pub,tot,rel,up.switchgames(i,na,pub,dev,tot,rel,up) \rightarrow developer(dev)$

$\forall pub,na.\ \exists i,dev,tot,rel,up.switchgames(i,na,pub,dev,tot,rel,up) \rightarrow hasPublished(pub, na)$

$\forall dev,na.\ \exists i,pub,tot,rel,up.switchgames(i,na,pub,dev,tot,rel,up) \rightarrow hasDeveloped(dev, na)$

MAPPING LAYER (3)

Finally, from **ps4games** we can take information about games, publishers and developers:

$\forall na,rel.\ \exists i,pub,dev,ge.ps4games(i,na,pub,rel,dev,ge) \rightarrow game(na,rel)$

$\forall pub.\ \exists i,na,rel,dev,ge.ps4games(i,na,pub,rel,dev,ge) \rightarrow publisher(pub)$

$\forall dev.\ \exists i,na,pub,rel,ge.ps4games(i,na,pub,rel,dev,ge) \rightarrow developer(dev)$

$\forall pub,na.\ \exists i,rel,dev,ge.ps4games(i,na,pub,rel,dev,ge) \rightarrow hasPublished(pub, na)$

$\forall dev,na.\ \exists i,pub,rel,ge.ps4games(i,na,pub,rel,dev,ge) \rightarrow hasDeveloped(dev, na)$

MAPPING LAYER (4)

In **awards**, we have information about the Game, People and Company categories.

When the award is given to a person or a company, the information about the winner is inside **gameawards**.

$$\forall y, cat, nom, win.(\exists com, v.gameawards(y,cat,nom,com,win,v) \wedge \wedge (\vee_{CAT \in PeopleCategory} cat=CAT)) \rightarrow \exists na.award(y,cat,nom,win,na,\text{"PEOPLE"})$$

$$\forall y, cat, nom, win.(\exists com, v.gameawards(y,cat,nom,com,win,v) \wedge \wedge (\vee_{CAT \in CompanyCategory} cat=CAT)) \rightarrow \exists na.award(y,cat,nom,win,na,\text{"COMPANY"})$$

MAPPING LAYER (5)

When the award is assigned to a game, we don't take the information about the winner from **gameawards**.
Instead we have to take the developer of the game associated to the award from the games datasets.

For **steamgames**, we have the following:

$$\forall y,cat,nom,win,dev.(\exists com, v, i, rel, rat, ge, pub.gameawards(y,cat,nom,com,win,v) \wedge \wedge (\bigvee_{CAT \in GameCategory} cat=CAT) \wedge steamgames(i,nom,rel,rat,ge,pub,dev)) \rightarrow award(y,cat,dev,win,nom,"STEAM")$$

MAPPING LAYER (6)

Consequently, for **switchgames** and **ps4games**, we identify the following mapping specifications:

$\forall y,cat,nom,win,dev.(\exists com, v, i, pub, tot, rel, up.gameawards(y,cat,nom,com,win,v) \wedge$
$\wedge (\bigvee_{CAT \in GameCategory} cat=CAT) \wedge switchgames(i,nom,pub,dev,tot,rel,up)) \rightarrow$
$award(y,cat,dev,win,nom,"SWITCH")$

$\forall y,cat,nom,win,dev.(\exists com, v, i, pub, rel, ge.gameawards(y,cat,nom,com,win,v) \wedge$
$\wedge (\bigvee_{CAT \in GameCategory} cat=CAT) \wedge ps4games(i,nom,pub,rel,dev,ge)) \rightarrow$
$award(y,cat,dev,win,nom,"PS4")$

MAPPING LAYER (7)

Finally, we identify the cross-platform games as being those games that exist on multiple platforms.
In particular, we have to take information from all the pairs of different possible games.

For **steamgames** and **switchgames** we have:

$$\forall na,rel.(\exists i_1,rat_1,ge_1,pub_1,dev_1,i_2,pub_2,dev_2,tot_2,rel_2,up_2.steamgames(i_1,na,rel,rat_1,ge_1,pub_1,dev$$
$$\wedge\ switchgames(i_2,na,pub_2,dev_2,tot_2,rel_2,up_2)) \rightarrow crossgame(na,rel)$$

MAPPING LAYER (8)

For **steamgames** and **ps4games** we have:

$$\forall na,rel.(\exists i_1,rat_1,ge_1,pub_1,dev_1,i_2,pub_2,rel_2,dev_2,ge_2.steamgames(i_1,na,rel,rat_1,ge_1,pub_1,dev_1) \wedge \wedge ps4games(i_2,na,pub_2,rel_2,dev_2,ge_2)) \rightarrow crossgame(na,rel)$$

For **ps4games** and **switchgames** we have:

$$\forall na,rel.(\exists i_1,pub_1,dev_1,ge_1,i_2,pub_2,dev_2,tot_2,rel_2,up_2.ps4games(i_1,na,pub_1,rel,dev_1,ge_1) \wedge \wedge switchgames(i_2,na,pub_2,dev_2,tot_2,rel_2,up_2)) \rightarrow crossgame(na,rel)$$

QUERY 1 and 2

1. find all the cross-platform games that have been published by a publisher company

   $\{(pub,na) \mid \exists\, rel.crossgame(na,rel) \wedge publisher(pub) \wedge hasPublished(pub,na)\}$

2. find all the publishers and their published games that won an award in any Game Category

   $\{(pub,na) \mid \bigvee_{CAT \in GameCategory} (\exists\, rel, y, nom, ty.publisher(pub) \wedge hasPublished(pub,na) \wedge$

   $\wedge\, game(na,rel) \wedge award(y,CAT,nom,"1",na,ty))\}$

QUERY 3 and 4

3. find all cross-platform games that won an award for both "Game of the Year" and "Best Action/Adventure Game"

$$\{(na) \mid \exists rel,y_1,nom_1,ty_1,y_2,nom_2,ty_2.award(y_1,\text{"Game of the Year"},nom_1,\text{"1"},na,ty_1) \wedge \wedge award(y_2,\text{"Best Action/Adventure Game"},nom_2,\text{"1"},na,ty_2) \wedge crossgame(na,rel)\}$$

4. find all STEAM games that won at least two awards in the same year in two different Game Categories

$$\{(na) \mid \bigvee_{CAT1 \in GameCategory, CAT2 \in GameCategory,\ CAT1 \neq CAT2} (\exists rel,y,nom_1,nom_2.game(na,rel) \wedge \wedge award(y,CAT_1,nom_1,\text{"1"},na,\text{"STEAM"}) \wedge award(y,CAT_2,nom_2,\text{"1"},na,\text{"STEAM"}))\}$$

QUERY 5 and 6

5. find all the people that won at least one award and the year in which they won the award in the People Category

$$\{(nom,y) \mid \bigvee_{CAT \in PeopleCategory} (\exists\, na, ty.award(y,CAT,nom,"1",na,ty))\}$$

6. find all the people in the People Category that have been nominated for an award for two different games

$$\{(nom) \mid \bigvee_{CAT \in PeopleCategory} (\exists\, y_1, win_1, na_1, ty_1, y_2, win_2, na_2, ty_2.\ \neg(na_1 = na_2) \wedge$$
$$\wedge\ award(y_1,CAT,nom,win_1,na_1,ty_1) \wedge award(y_2,CAT,nom,win_2,na_2,ty_2))\}$$

QUERY 7

7. find all the developers who won an award as best developer of the year and best game developed (we consider any Game Category)

$$\{(nom) \mid \bigvee_{CAT \in GameCategory} (\exists\, rel,y_1,na_1,ty_1,y_2,na_2,ty_2.game(na_1,rel) \wedge developer(nom) \wedge$$

$$\wedge\, hasDeveloped(nom,na_1) \wedge award(y_1,CAT,nom,"1",na_1,ty_1) \wedge$$

$$\wedge\, award(y_2,"Developer\ of\ the\ Year",nom,"1",na_2,ty_2))\}$$

3

IMPLEMENTATION

Pre-processing,
Mapping Transformations,
Materialization

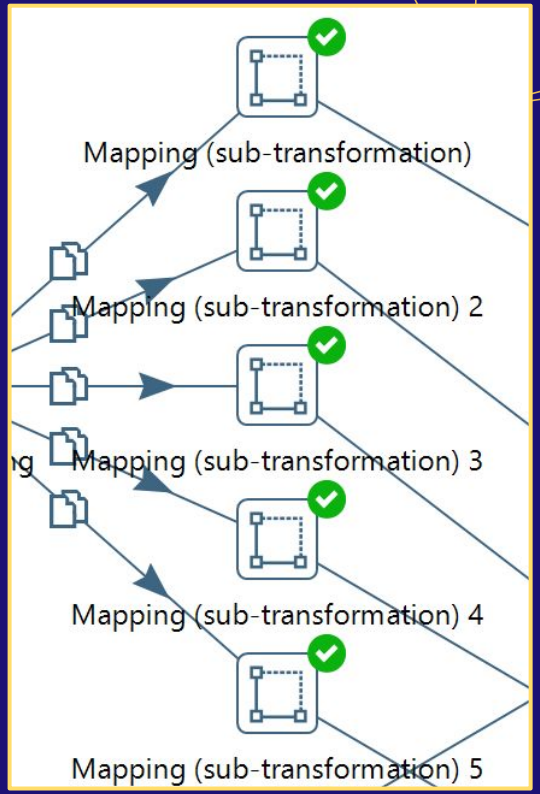
PRE-PROCESSING (PENTAHO)

After the data has been loaded from a CSV file, all the **pre-processing** is performed directly in Pentaho:

- “pre-process” eliminates useless columns
- “Regex game”, “Regex publisher” and “Regex developer” match with non UTF-8 encoded characters and “Filter rows” eliminates those tuples that are identified with the regular expressions (circa 200 tuples)
- “Replace in string” does the substitution of problematic characters

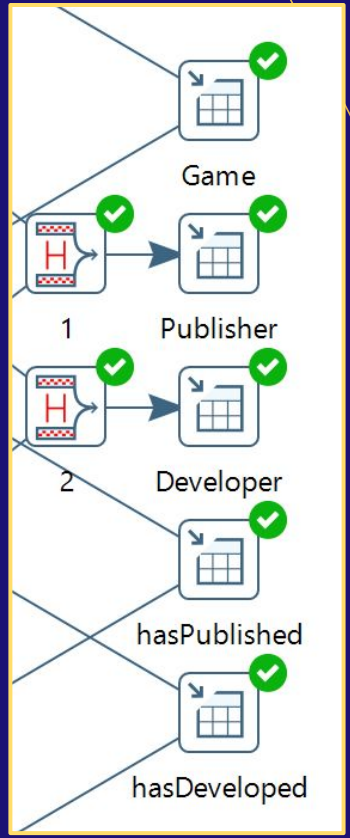
MAPPING TRANSFORMATIONS

- **Mapping transformations** are used in Pentaho to perform the transformations of the Mapping Layer
- These mappings call other transformations that actually perform the mapping (they are sub-transformations)
- In the image are illustrated the five mapping transformations corresponding to the mappings applied to “steamgames”



MATERIALIZATION

- The output of the mapping transformation is **materialized** as an incomplete database
- After connecting to a database (PostgreSQL in the project) several mapping outputs coming from different transformations form the atoms of that database
- For specific tables, as the award and crossgame tables, a merge join is performed before the materialization to define the more complicated mappings



REFERENCES FOR THE PRESENTATION

The datasets used in this presentation had been taken and reworked from the following sources:

- **Steam Games**: https://www.kaggle.com/datasets/whigmalwhim/steam-releases
- **Switch Games**: https://www.kaggle.com/datasets/uadithyan/nintendo-switch-games
- **PS4 Games**: https://www.kaggle.com/datasets/shivamb/all-playstation-4-games
- **The Game Awards**: https://www.kaggle.com/datasets/unanimad/the-game-awards

THANK YOU